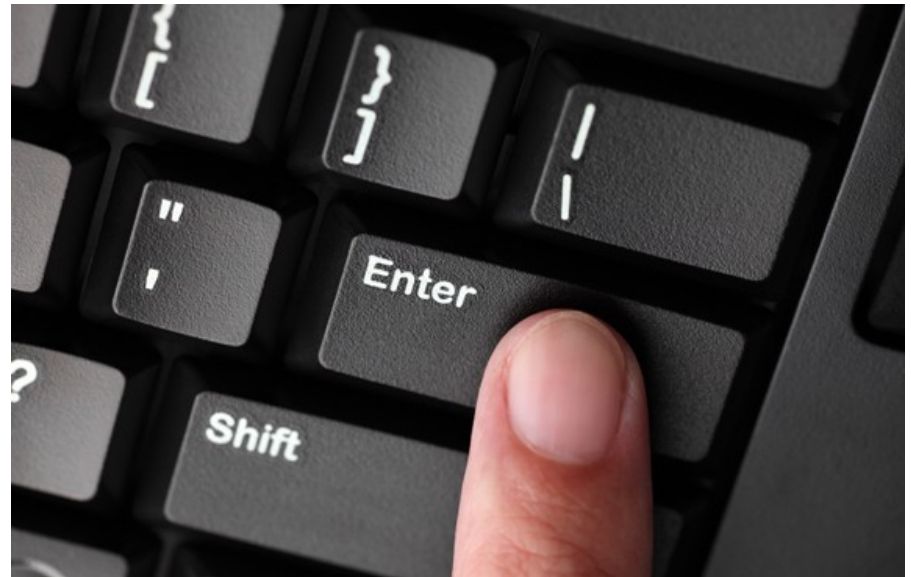


# Eventos con jQuery

## Recordando: Eventos

---

Los eventos comunican acciones realizadas tanto por el navegador como por el usuario, y ayudan a mejorar la interacción entre una persona y un sitio web.



# Eventos en jQuery



## Eventos con jQuery

---

Los eventos siguen siendo los mismos usando JavaScript puro o jQuery, ya que el evento en sí (por ejemplo click) no cambia, **sólo cambia el modo de escribirlo.**

### HTML

```
<div id="myDiv">Some Content</div>
```

### jQuery

```
$('#myDiv').click(function(){  
    //Some code  
});
```

### HTML

```
<div id="myDiv" onClick="divFunction()">Some Content</div>
```

### JavaScript function call

```
function divFunction(){  
    //Some code  
}
```

# Eventos comunes



Eventos comunes mouse: .click() y .dblclick()

---

**.click():** La función se ejecuta cuando el usuario hace clic en el elemento HTML.

```
$(document).ready(function(){  
    $("p").click(function(){  
        $(this).hide();  
    });  
});
```

**.dblclick():** La función se ejecuta cuando el usuario hace doble clic en el elemento HTML.

```
$(document).ready(function(){  
    $("p").dblclick(function(){  
        $(this).hide();  
    });  
});
```

Eventos comunes mouse: .mouseenter() y .mouseleave()

---

**.mouseenter():** Se produce cuando el puntero del ratón está sobre el elemento seleccionado.

**.mouseleave():** Se produce cuando el puntero del ratón sale del elemento seleccionado.

```
$(document).ready(function(){  
    $("p").mouseenter(function(){  
        $("p").css("background-color", "yellow");  
    });  
    $("p").mouseleave(function(){  
        $("p").css("background-color", "lightgray");  
    });  
});
```

Eventos comunes teclado: .keyDown, .keyPress, .keyUp

---

**.keydown():** Se ejecuta cuando se pulsa la tecla, comienzo de la presión.

**.keypress():** Se ejecuta cuando la tecla está presionada.

**.keyup():** Se ejecuta cuando la tecla es soltada.

```
$(document).ready(function(){
    $("input").keydown(function(){
        $("input").css("background-color", "yellow");
    });
    $("input").keyup(function(){
        $("input").css("background-color", "pink");
    });
});
```

```
i = 0;
$(document).ready(function(){
    $("input").keypress(function(){
        $("span").text(i += 1);
    });
});
```



## Eventos comunes teclado: notas

---

### Notas:

-El evento de *keypress()* es similar al evento *keydown()*, sin embargo, el evento de *keypress()* no se dispara para todas las teclas (por ejemplo ALT, CTRL, SHIFT, ESC).

Eventos comunes Forms: .submit(), .change(),

---

**.submit():** Sucede cuando el formulario es enviado. Sólo se puede utilizar con elementos form.

```
$(document).ready(function(){  
    $("form").submit(function(){  
        alert("Submitted");  
    });  
});
```

**.change():** Sucede cuando cambia el valor de un elemento.  
(Sólo se puede utilizar con elementos input, textarea y select).

```
$(document).ready(function(){  
    $("input").change(function(){  
        alert("The text has been changed.");  
    });  
});
```

Eventos no tan  
comunes




Eventos no tan comunes: `.off()` y `.contextmenu()`

---

**.off():** Sirve para desvincular eventos. Por ejemplo para eliminar el evento click en a podríamos hacer lo siguiente:

```
$("a").off("click");
```

**.contextmenu():** Se activa cuando se hace click en el botón derecho del mouse (antes de que aparezca el menú contextual), o cuando se pulsa la tecla de menú contextual (tecla con símbolo ).

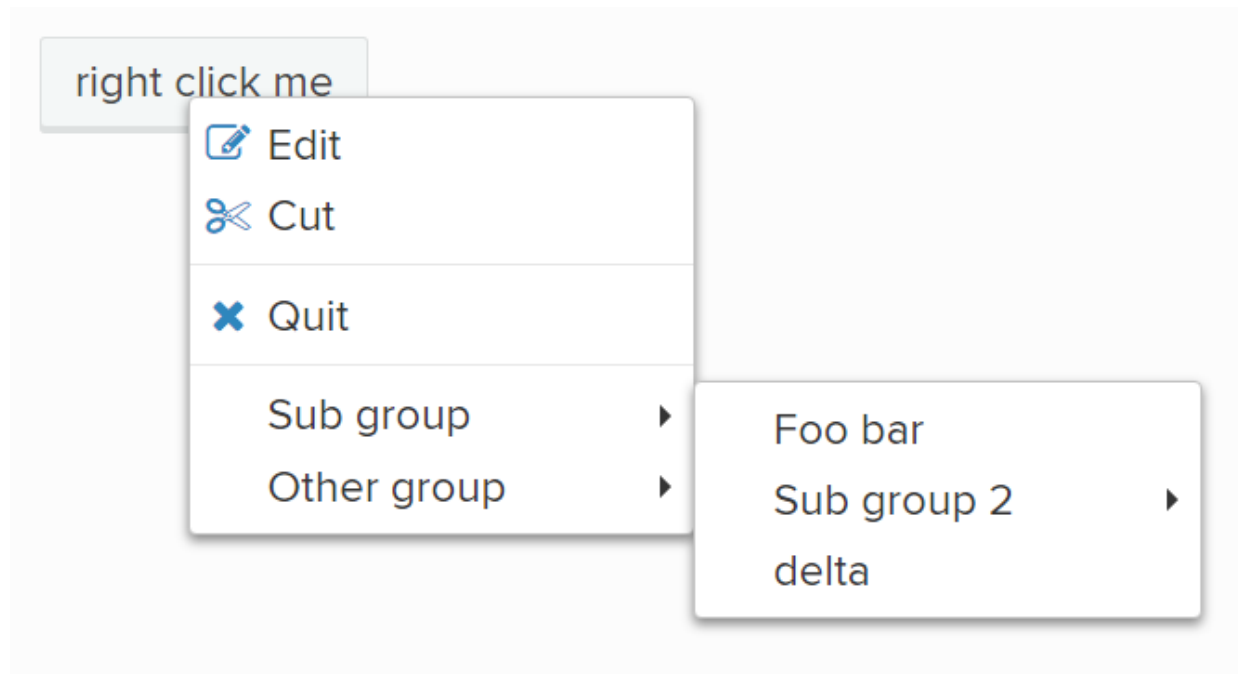
Podemos probar el funcionamiento del evento a  
quí.

Eventos comunes teclado: notas

---

## ¿Qué es un menú contextual?

El menú contextual es un menú de opciones que aparece al hacer clic derecho sobre un elemento.



Eventos no tan comunes: `.hover()`

---

**.hover():** Especifica dos funciones a ejecutar cuando el puntero del ratón pasa sobre los elementos seleccionados.

Este método activa tanto el evento `MouseEnter` y `MouseLeave`.

**Podemos probar el funcionamiento del evento aquí**

Eventos no tan comunes: `.event.pageX` y `.event.pageY`

---

**.event.pageX:** Devuelve la posición del puntero del ratón , con relación al borde izquierdo del documento.



**.event.pageY:** Devuelve la posición del puntero del ratón , con relación al borde superior del documento.



Eventos no tan comunes: `.event.target` y `.event.type`

---

**.event.target:** Devuelve qué elemento DOM ha activado el evento.

```
$("body").click(function(event) {  
    $("#log").html("clicked: " + event.target.nodeName);  
});
```

**.event.type:** Devuelve qué tipo de evento se desencadenó.

```
$("a").click(function(event) {  
    alert(event.type); // click  
});
```



Eventos no tan comunes: `.select()`

---

**`.select()`**: El evento de selección se produce cuando se selecciona un texto (marcado) en un área de texto o un campo de texto

El método de `select( )` activa el evento de selección, o atribuye una función a ejecutar cuando se produce un evento `select` .

```
1 | $( "#other" ).click(function() {  
2 |     $( "#target" ).select();  
3 | });
```

# Especiales



## Especiales: hide y show

**.hide():** Oculta elementos HTML.

**.show():** Muestra los elementos.

```
$(selector).hide(speed, callback);
```

```
$(selector).show(speed, callback);
```

```
<script>
$(document).ready(function(){
    $("#hide").click(function(){
        $("p").hide();
    });
    $("#show").click(function(){
        $("p").show();
    });
});
</script>
```

- El parámetro de velocidad *speed* (opcional) especifica la velocidad de la ocultación/muestra, y puede tomar los siguientes valores: **slow**, **fast** o **milliseconds**.
- El parámetro *callback* (opcional) es una función a ejecutar después de que el método hide o show se completa.

## Especiales: toggle

---

**.toggle():** Alterna entre los métodos `hide()` y `show()`.

Oculto elementos que se ven y muestra elementos ocultos.



\$

`(selector).toggle(speed, callback);`

El parámetro de velocidad *speed* (opcional) puede tomar los siguientes valores: `slow`, `fast` o `milliseconds`.

El parámetro `callback` (opcional) es una función a ejecutar después de completar el evento `toggle()`

```
$(document).ready(function(){  
    $("button").click(function(){  
        $("p").toggle();  
    });  
});
```

Especiales: `.fadeIn()` y `.fadeOut()`

---

**.fadeIn():** Usado para aparecer en un elemento oculto.

**.fadeOut():** Usado para desaparecer un elemento visible.

```
$(selector).fadeIn(speed, callback);  
$(selector).fadeOut(speed, callback);
```

```
$(document).ready(function(){  
    $("button").click(function(){  
        $("#div1").fadeIn();  
        $("#div2").fadeIn("slow");  
        $("#div3").fadeIn(3000);  
    });  
});
```

```
$(document).ready(function(){  
    $("button").click(function(){  
        $("#div1").fadeOut();  
        $("#div2").fadeOut("slow");  
        $("#div3").fadeOut(3000);  
    });  
});
```

- El parámetro de velocidad *speed* (opcional) especifica la duración del efecto. Puede tomar los siguientes valores: *slow*, *fast* o *milliseconds*.
- El parámetro *callback* (opcional) es una función que se ejecuta cuando el efecto se haya completado.

## Especiales: .fadeToggle()

**.fadeToggle():** Alterna entre el métodos fadeIn y fadeOut.

- El parámetro de velocidad *speed* (opcional) especifica la duración del efecto. Puede tomar los siguientes valores: slow, fast o miliseconds.
- El parámetro *callback* (opcional) es una función que se ejecutará cuando el efecto se haya completado.

```
$(selector).fadeToggle(speed,callback);
```

```
$(document).ready(function(){
    $("button").click(function(){
        $("#div1").fadeToggle();
        $("#div2").fadeToggle("slow");
        $("#div3").fadeToggle(3000);
    });
});
```

Especiales: .fadeTo()

**.fadeTo():** Permite la decoloración a una opacidad dada (valor entre 0 y 1).

```
$(document).ready(function(){  
    $("button").click(function(){  
        $("#div1").fadeTo("slow", 0.15);  
        $("#div2").fadeTo("slow", 0.4);  
        $("#div3").fadeTo("slow", 0.7);  
    });  
});
```

**`$(selector).fadeTo(speed,opacity,callback);`**

- El parámetro de velocidad opcional especifica la duración del efecto. Puede tomar los siguientes valores: slow, fast o milliseconds.
- El parámetro de opacidad especifica el nivel de opacidad que queremos asignar al elemento, el valor deberá estar entre 0.00 y 1.00.
- El parámetro callback opcional es una función a ejecutar después de que el efecto se haya completado.

## Especiales: slideDown()

**.slideDown():** Desliza hacia abajo los elementos seleccionados.

```
$(selector).slideDown(speed,easing,callback)
```

```
$(document).ready(function(){  
    $(".btn1").click(function(){  
        $("p").slideUp();  
    });  
    $(".btn2").click(function(){  
        $("p").slideDown();  
    });  
});
```

- Speed especifica la duración del efecto. Puede tomar los valores: slow, fast o milliseconds.
- Easing puede recibir los valores linear (velocidad lineal) y swing (lento al comienzo y al final, y rápido al medio)
- El parámetro callback (opcional). Es una función que se ejecutará cuando el efecto



## Especiales: slideUp()

**.slideUp():** Oculta/desliza hacia arriba los elementos seleccionados.

```
$(selector).slideUp(speed,easing,callback)
```

```
<script>
$(document).ready(function(){
    $(".btn1").click(function(){
        $("p").slideUp(1000);
    });
    $(".btn2").click(function(){
        $("p").slideDown(1000);
    });
});
```

- Speed especifica la duración del efecto. Puede tomar los valores: "slow", "fast", o miliseconds.
- Easing puede recibir los valores "linear" (velocidad lineal) y "swing" (lento al comienzo y al final, y rápido al medio)
- El parámetro callback (opcional) es una función a ejecutar cuando el efecto se haya

Especiales: slideToggle()

**.slideToggle():** Alterna entre los métodos slideIn() y slideOut()

```
$(selector).fadeToggle(speed,easing,callback)
```

```
$(document).ready(function(){  
    $("button").click(function(){  
        $("#div1").fadeToggle();  
        $("#div2").fadeToggle("slow");  
        $("#div3").fadeToggle(3000);  
    });  
});
```

# Eventos vs. On



## Eventos vs. On

En la última versión de jQuery han intentado unificar las APIs de manejo de eventos en los métodos **on** y **off**. Estos métodos sustituyen a los antiguos `bind()`, `delegate()` y `live()`.

Si comparamos el uso de `on()` con el de atajos que ya existían, como `click()`, tenemos el siguiente código:

```
1 // Asignar un manejador de eventos usando click
2 $("#header a").click(function() {
3     // manejar el evento
4 });
5
6 // Asignar un manejador de eventos usando on
7 $("#header a").on("click", function() {
8     // manejar el evento
9 });|
```

## Eventos vs. On

---

Estas dos construcciones son equivalentes, pero `on()` permite hacer muchas más cosas.

Una de las cosas más interesantes es que **permite usar un mismo manejador de eventos para múltiples elementos html suscribiendo el manejador a un elemento padre.**

## Eventos vs. On

### Ejemplo:

```
1 <div id="content">
2     <a href="#">enlace1</a>
3     <a href="#">enlace2</a>
4     <a href="#">enlace2</a>
5 </div>
```

Si por algún motivo queremos asignar el mismo manejador de eventos a todos los elementos `<a>`, podemos hacerlo de la siguiente forma:

```
1 $("#content").on("click", "a", function() {
2     // manejar el evento
3     // $(this) apunta al <a> que ha generado el evento
4 });|
```

## Eventos vs. On: Ventajas.

---

- **Sólo se crea una única función**, independientemente del número de elementos `<a>` que tengamos, reduciendo el consumo de recursos.
- **Es válido para elementos que no existen todavía.** Si apareciera un nuevo elemento `<a>` dentro del `<div>`, automáticamente estaríamos manejando su evento click. Esto es especialmente útil cuando generamos html dinámicamente.

## Diferencias entre bind, live, delegate y on

---

Cronológicamente, la primera en aparecer fue **bind**. Después apareció **live** para cubrir lo que no llenaba bind (la delegación de eventos), pero live no funcionaba como se esperaba, así que incorporaron **delegate** en detrimento de live.

Por último y **para evitar que la gente se volviera loca teniendo que decidir si utilizar bind, live o delegate, surgió **on****, que en un solo método reúne la funcionalidad de todos los métodos anteriores.



.on()

---

**.on():** La opción más recomendable. Se ha diseñado para substituir los eventos `delegate()` y `bind()`. Si estamos usando `bind()` pasar a usar `on()` es tan fácil como cambiar las etiquetas, una por la otra.

Este método **cede al controlador de eventos del documento a los nuevos elementos que coincidan con los criterios de nuestra condición**. Con el fin de mejorar su eficiencia, `on()` debe unirse de forma específica, no generalizada.

.on()

---

La diferencia más notable con `live()` es que con **`on()`** sólo realiza una llamada en vez de las *n llamadas* que realiza `live()`. Dicho de otra forma, cuando hacemos click en un elemento, si lo manejamos con `on()` sólo se hará una llamada al DOM desde ese elemento.

Si manejamos el evento click con `live()` se harán tantas llamadas como elementos haya en la página en la que estamos trabajando.